# (22F) CST8277 Enterprise Application Programming

# Assignment 2: DataBank-JSF-JPA-EJB-BootStrap

Please read this document carefully. If your submission does not meet the requirements as stated here, you may lose marks even though your program runs. Additionally, this assignment is also a teaching opportunity, that is **materials presented in this document may be on the midterm and/or the final exam!**

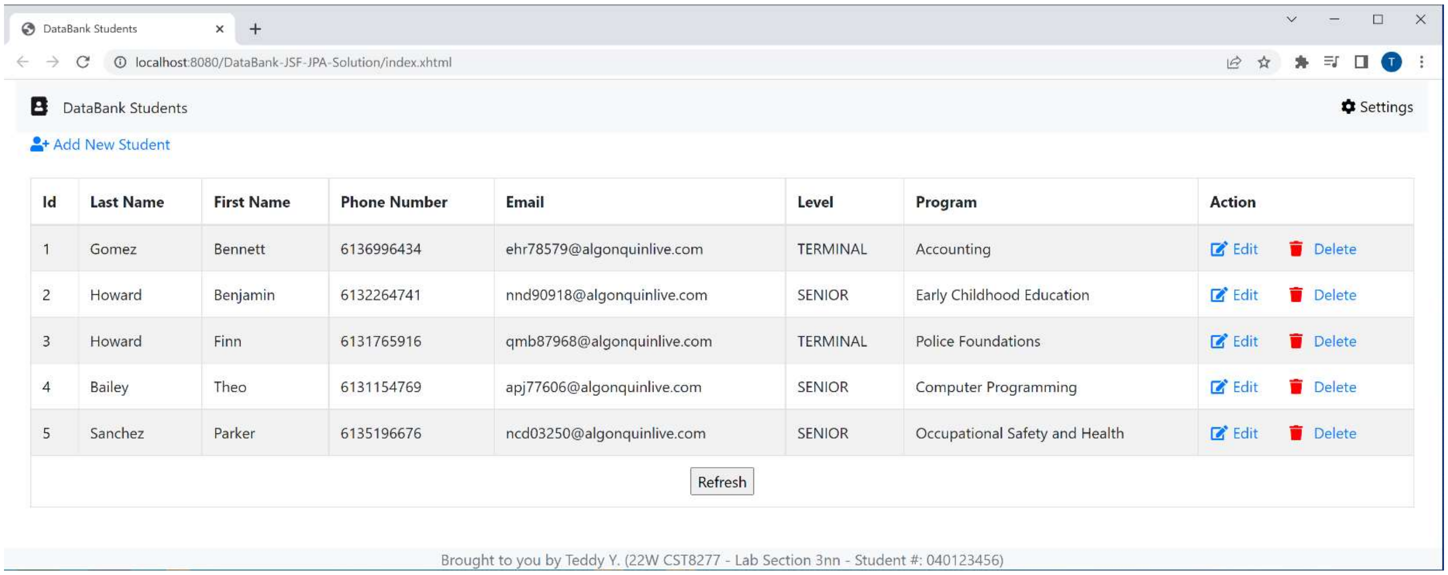
Assignment 2's submission is to be uploaded to the Assignment 2 Brightspace submission folder. It does not matter if it is a zip file containing a zip file containing... many-levels-of-nested-zip-files or if you upload the required artifacts individually, Brightspace will handle things. If you have a 100% perfect solution but it is not in the correct folder, you will receive a mark of zero ☹ If you know that you made mistake, please upload it to the correct folder ASAP.

## Overview

For Assignment 2, ACME Corp. has tasked you to improve the Student Directory web application you developed back in Assignment 1. Specifically, the company wants you to:

1. Use Java Persistence API (JPA) to access the database and provide the required CRUD capabilities (instead of directly using JDBC)
2. Use Bootstrap CSS framework to layout and style the Student Directory web application as a Single Page Application (SPA)

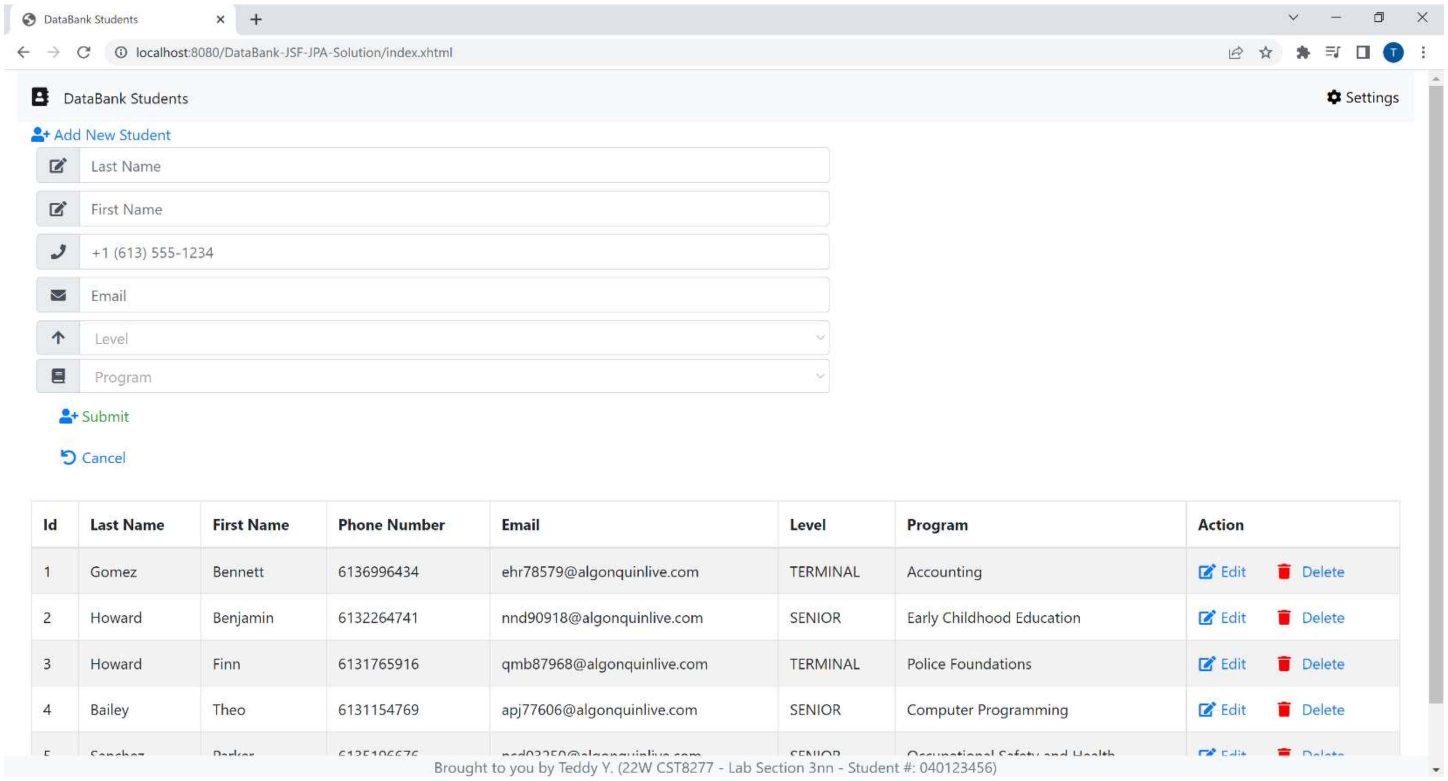
The main page of the improved web application is shown below. The main page shows the list of students already on the database.



| Id | Last Name | First Name | Phone Number | Email | Level | Program | Action |
|---|---|---|---|---|---|---|---|
| 1 | Gomez | Bennett | 6136996434 | ehr78579@algonquinlive.com | TERMINAL | Accounting | Edit Delete |
| 2 | Howard | Benjamin | 6132264741 | nnd90918@algonquinlive.com | SENIOR | Early Childhood Education | Edit Delete |
| 3 | Howard | Finn | 6131765916 | qmb87968@algonquinlive.com | TERMINAL | Police Foundations | Edit Delete |
| 4 | Bailey | Theo | 6131154769 | apj77606@algonquinlive.com | SENIOR | Computer Programming | Edit Delete |
| 5 | Sanchez | Parker | 6135196676 | ncd03250@algonquinlive.com | SENIOR | Occupational Safety and Health | Edit Delete |

Brought to you by Teddy Y. (22W CST8277 - Lab Section 3nn - Student #: 040123456)

Clicking the “Add New Student” button on the main page would show the Add New Student form while staying on the main page:

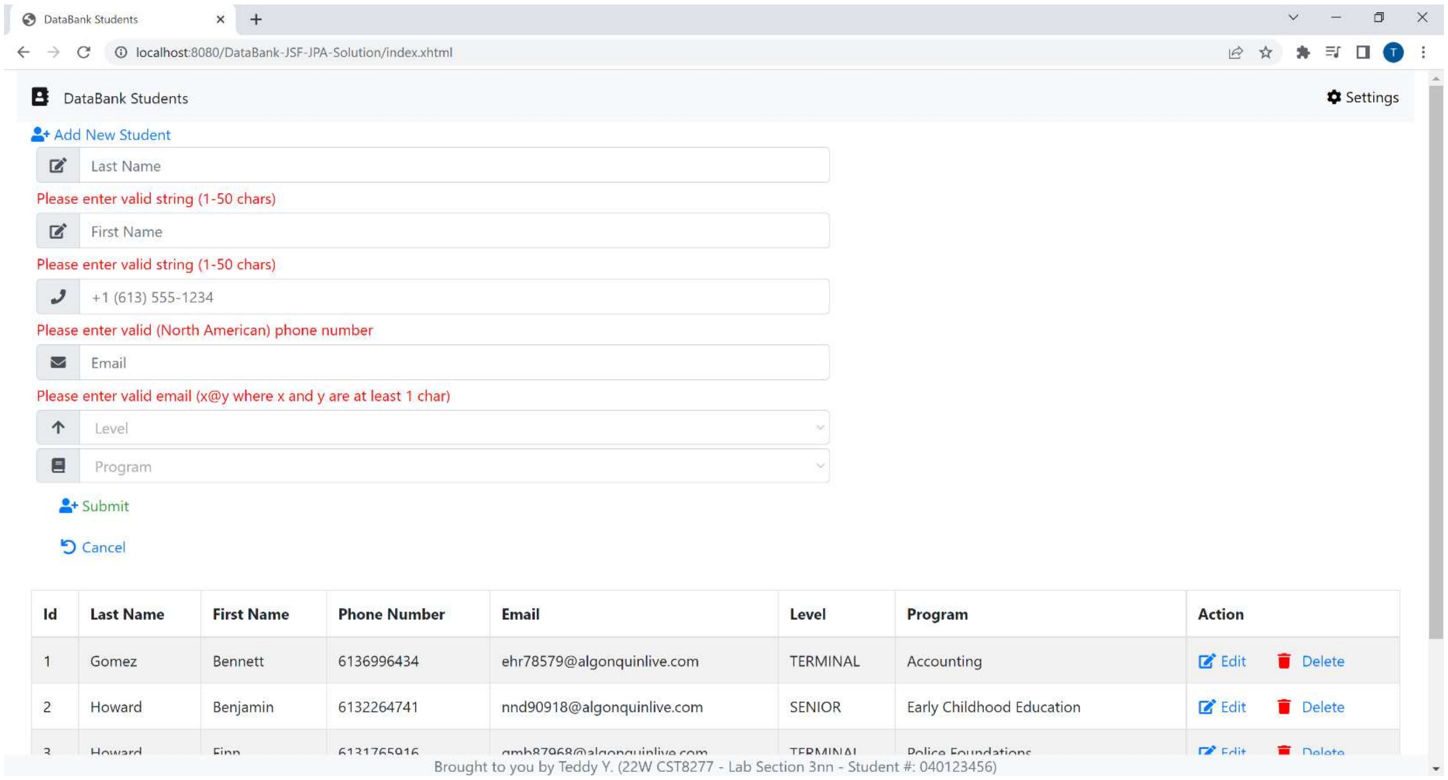


| Id | Last Name | First Name | Phone Number | Email | Level | Program | Action |
|---|---|---|---|---|---|---|---|
| 1 | Gomez | Bennett | 6136996434 | ehr78579@algonquinlive.com | TERMINAL | Accounting | Edit Delete |
| 2 | Howard | Benjamin | 6132264741 | nnd90918@algonquinlive.com | SENIOR | Early Childhood Education | Edit Delete |
| 3 | Howard | Finn | 6131765916 | qmb87968@algonquinlive.com | TERMINAL | Police Foundations | Edit Delete |
| 4 | Bailey | Theo | 6131154769 | apj77606@algonquinlive.com | SENIOR | Computer Programming | Edit Delete |

Brought to you by Teddy Y. (22W CST8277 - Lab Section 3nn - Student #: 040123456)

Clicking the “Submit” button (after entering all the necessary information) or “Cancel” button will just hide the Add New Student form and add the new student to the table (if applicable).

Note that if the user didn’t provide all the necessary information in the correct format, validation error messages will be displayed as shown below:



Please enter valid string (1-50 chars)

Please enter valid string (1-50 chars)

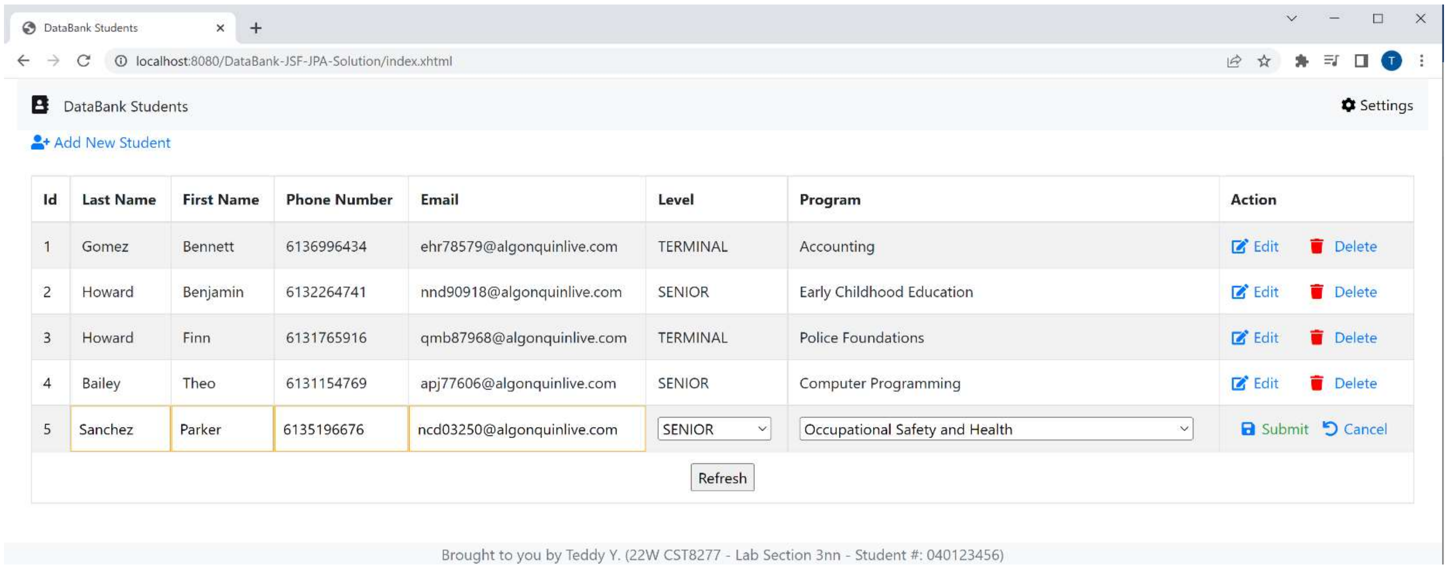
Please enter valid (North American) phone number

Please enter valid email (x@y where x and y are at least 1 char)

| Id | Last Name | First Name | Phone Number | Email | Level | Program | Action |
|---|---|---|---|---|---|---|---|
| 1 | Gomez | Bennett | 6136996434 | ehr78579@algonquinlive.com | TERMINAL | Accounting | Edit Delete |
| 2 | Howard | Benjamin | 6132264741 | nnd90918@algonquinlive.com | SENIOR | Early Childhood Education | Edit Delete |

Brought to you by Teddy Y. (22W CST8277 - Lab Section 3nn - Student #: 040123456)

Clicking the “Edit” button on the main page should enable the fields on the corresponding row for editing as shown below:



DataBank Students

localhost:8080/DataBank-JSF-JPA-Solution/index.xhtml

DataBank Students | Settings

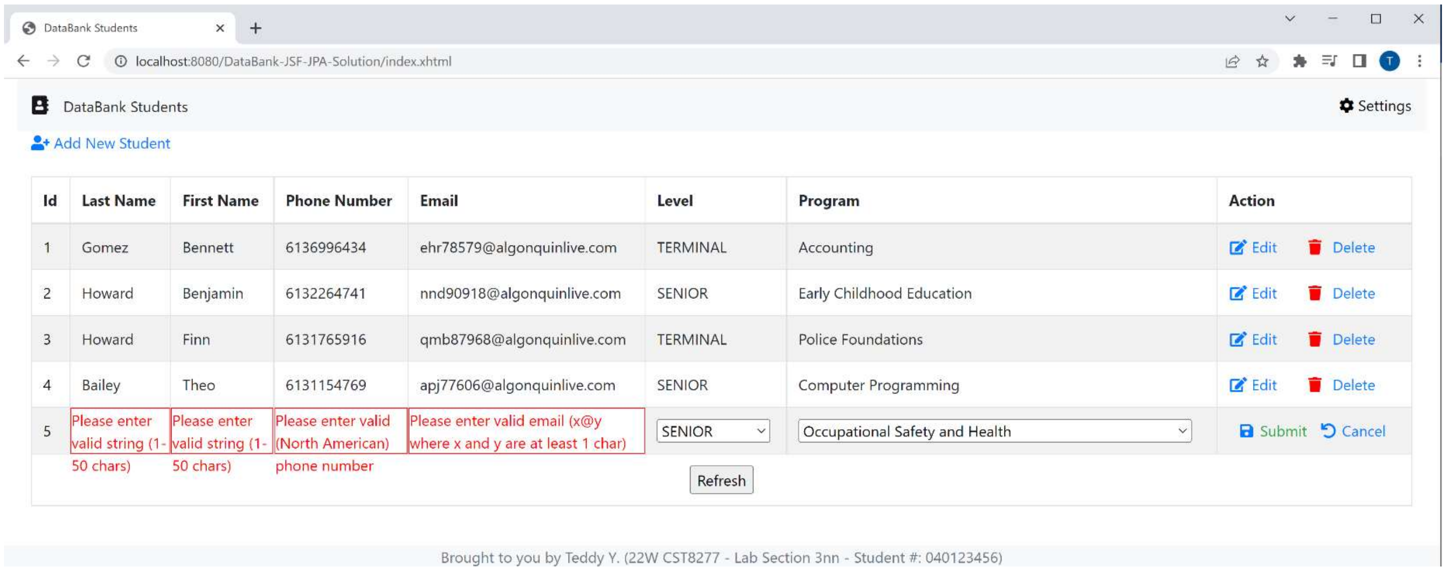
Add New Student

| Id | Last Name | First Name | Phone Number | Email | Level | Program | Action |
|---|---|---|---|---|---|---|---|
| 1 | Gomez | Bennett | 6136996434 | ehr78579@algonquinlive.com | TERMINAL | Accounting | Edit Delete |
| 2 | Howard | Benjamin | 6132264741 | nnd90918@algonquinlive.com | SENIOR | Early Childhood Education | Edit Delete |
| 3 | Howard | Finn | 6131765916 | qmb87968@algonquinlive.com | TERMINAL | Police Foundations | Edit Delete |
| 4 | Bailey | Theo | 6131154769 | apj77606@algonquinlive.com | SENIOR | Computer Programming | Edit Delete |
| 5 | Sanchez | Parker | 6135196676 | ncd03250@algonquinlive.com | SENIOR | Occupational Safety and Health | Submit Cancel |

Refresh

Brought to you by Teddy Y. (22W CST8277 - Lab Section 3nn - Student #: 040123456)

Similar to the add scenario, if the user didn't provide all the necessary information in the correct format, validation error message will be displayed as shown below:



DataBank Students

localhost:8080/DataBank-JSF-JPA-Solution/index.xhtml

DataBank Students | Settings

Add New Student

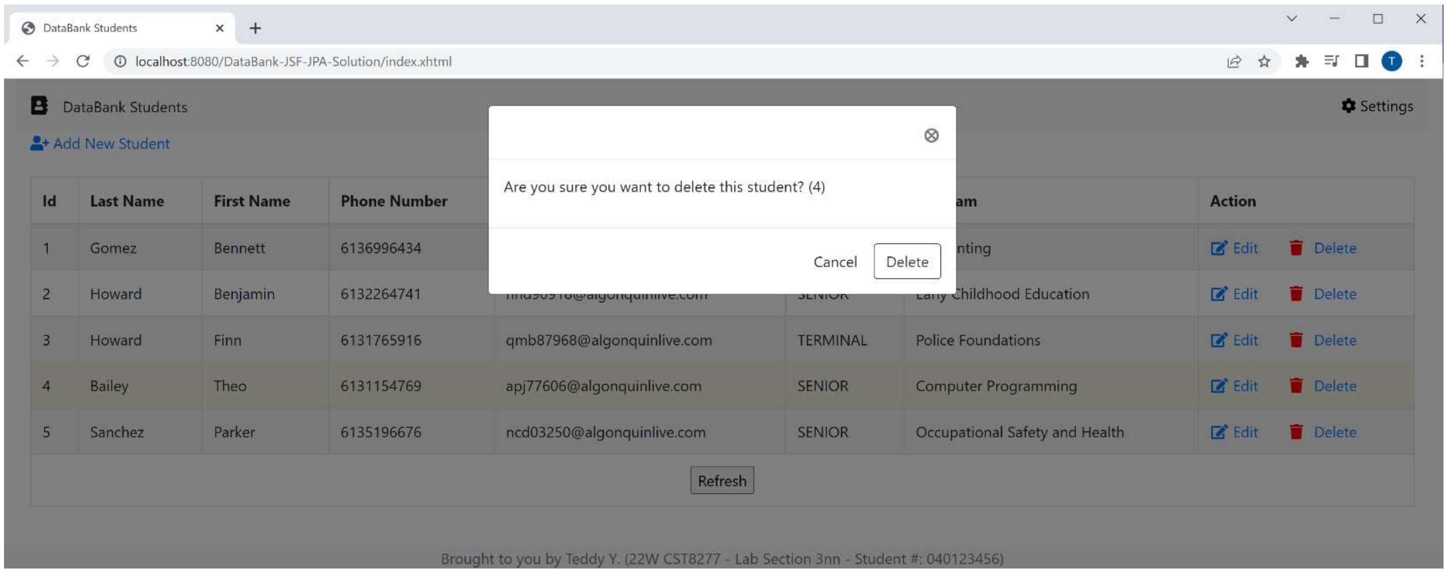
| Id | Last Name | First Name | Phone Number | Email | Level | Program | Action |
|---|---|---|---|---|---|---|---|
| 1 | Gomez | Bennett | 6136996434 | ehr78579@algonquinlive.com | TERMINAL | Accounting | Edit Delete |
| 2 | Howard | Benjamin | 6132264741 | nnd90918@algonquinlive.com | SENIOR | Early Childhood Education | Edit Delete |
| 3 | Howard | Finn | 6131765916 | qmb87968@algonquinlive.com | TERMINAL | Police Foundations | Edit Delete |
| 4 | Bailey | Theo | 6131154769 | apj77606@algonquinlive.com | SENIOR | Computer Programming | Edit Delete |
| 5 | Please enter valid string (1-50 chars) | Please enter valid string (1-50 chars) | Please enter valid (North American) phone number | Please enter valid email (x@y where x and y are at least 1 char) | SENIOR | Occupational Safety and Health | Submit Cancel |

Refresh

Brought to you by Teddy Y. (22W CST8277 - Lab Section 3nn - Student #: 040123456)

Clicking the "Cancel" button while editing will cancel the update operation and will just make the fields on the corresponding row back to being read-only with their original values.

Clicking the "Delete" button on the main page should trigger a pop-up to confirm if the user wants to proceed with deleting the selected student. Clicking OK on the pop-up should remove the selected student from the database as well as from the list of students shown on the main page. Clicking the "Cancel" button should simply cancel the delete operation.


DataBank Students

localhost:8080/DataBank-JSF-JPA-Solution/index.xhtml

DataBank Students

Settings

Add New Student

Are you sure you want to delete this student? (4)

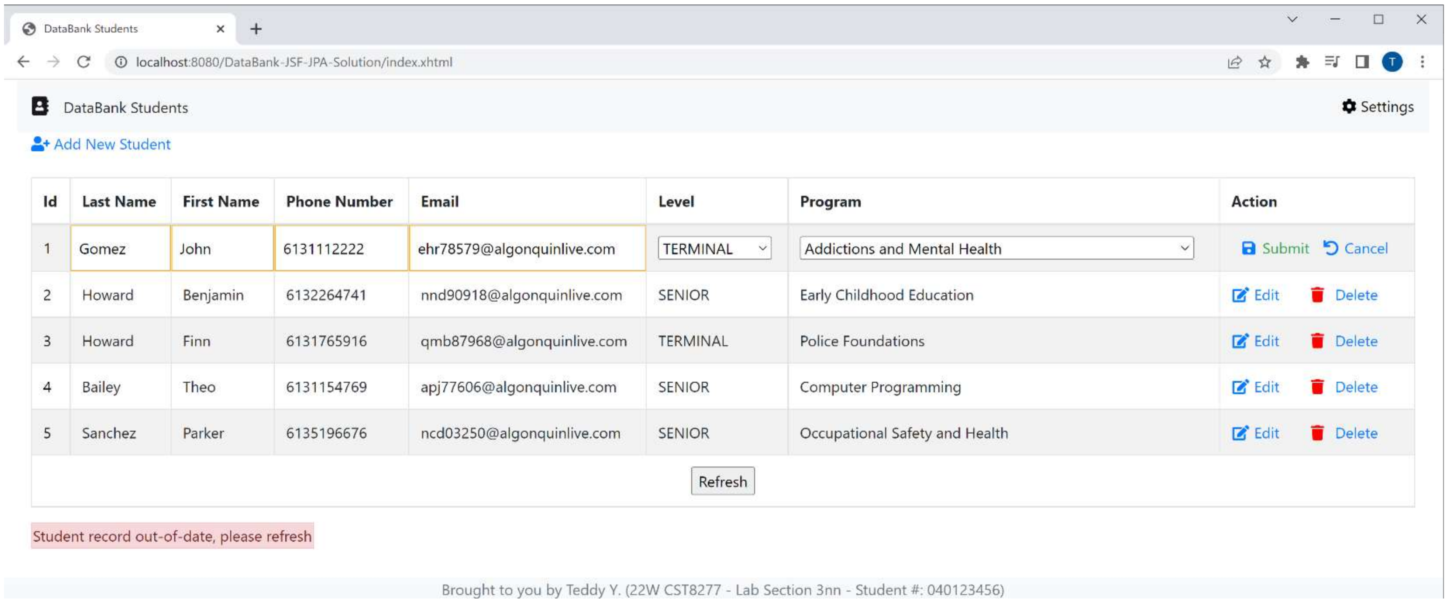
Cancel Delete

| Id | Last Name | First Name | Phone Number | | | am | Action |
|---|---|---|---|---|---|---|---|
| 1 | Gomez | Bennett | 6136996434 | | | nting | Edit Delete |
| 2 | Howard | Benjamin | 6132264741 | | | Childhood Education | Edit Delete |
| 3 | Howard | Finn | 6131765916 | qmb87968@algonquinlive.com | TERMINAL | Police Foundations | Edit Delete |
| 4 | Bailey | Theo | 6131154769 | apj77606@algonquinlive.com | SENIOR | Computer Programming | Edit Delete |
| 5 | Sanchez | Parker | 6135196676 | ncd03250@algonquinlive.com | SENIOR | Occupational Safety and Health | Edit Delete |

Refresh

Brought to you by Teddy Y. (22W CST8277 - Lab Section 3nn - Student #: 040123456)

Finally, if the user is trying to update a student that has already been updated by another user (e.g. from a different browser accessing the same web application) **OR** if the user is trying to update a student that has already been deleted by another user (e.g. from a different browser accessing the same web application), user-friendly error messages should be displayed as can be seen on the 2 screen shots below:


DataBank Students

localhost:8080/DataBank-JSF-JPA-Solution/index.xhtml

DataBank Students

Settings

Add New Student

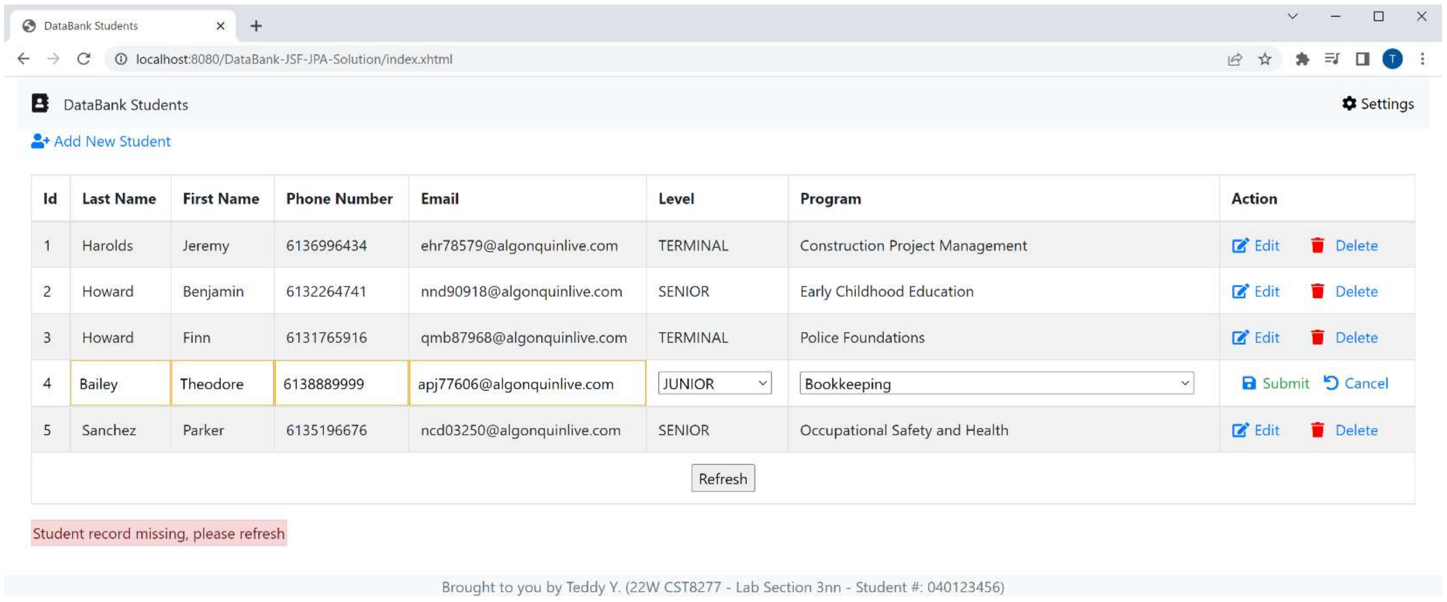
| Id | Last Name | First Name | Phone Number | Email | Level | Program | Action |
|---|---|---|---|---|---|---|---|
| 1 | Gomez | John | 6131112222 | ehr78579@algonquinlive.com | TERMINAL | Addictions and Mental Health | Submit Cancel |
| 2 | Howard | Benjamin | 6132264741 | nnd90918@algonquinlive.com | SENIOR | Early Childhood Education | Edit Delete |
| 3 | Howard | Finn | 6131765916 | qmb87968@algonquinlive.com | TERMINAL | Police Foundations | Edit Delete |
| 4 | Bailey | Theo | 6131154769 | apj77606@algonquinlive.com | SENIOR | Computer Programming | Edit Delete |
| 5 | Sanchez | Parker | 6135196676 | ncd03250@algonquinlive.com | SENIOR | Occupational Safety and Health | Edit Delete |

Refresh

Student record out-of-date, please refresh

Brought to you by Teddy Y. (22W CST8277 - Lab Section 3nn - Student #: 040123456)

User is trying to update a student that has already been updated by another user



| Id | Last Name | First Name | Phone Number | Email | Level | Program | Action |
|---|---|---|---|---|---|---|---|
| 1 | Harolds | Jeremy | 6136996434 | ehr78579@algonquinlive.com | TERMINAL | Construction Project Management | Edit Delete |
| 2 | Howard | Benjamin | 6132264741 | nnd90918@algonquinlive.com | SENIOR | Early Childhood Education | Edit Delete |
| 3 | Howard | Finn | 6131765916 | qmb87968@algonquinlive.com | TERMINAL | Police Foundations | Edit Delete |
| 4 | Bailey | Theodore | 6138889999 | apj77606@algonquinlive.com | JUNIOR | Bookkeeping | Submit Cancel |
| 5 | Sanchez | Parker | 6135196676 | ncd03250@algonquinlive.com | SENIOR | Occupational Safety and Health | Edit Delete |

Student record missing, please refresh

Brought to you by Teddy Y. (22W CST8277 - Lab Section 3nn - Student #: 040123456)

User is trying to update a student that has already been deleted by another user

## Theme for Assignment 2

After completing this assignment, you will have achieved the following:

1. Use JSF in conjunction with new EE components: EJBs and JPA
2. Create a model class for JPA with all appropriate mappings/annotations
3. Use a JSF `@ViewScoped` managed bean
4. Add validation to JSF views
5. Use Bootstrap CSS framework to layout and style the Student Directory application as a SPA

## Importing Assignment 2

ACME Corp. has provided you with a starter code ***DataBank-JSF-JPA-Skeleton.zip*** to complete. You must use the file ***DataBank-JSF-JPA-Skeleton.zip*** from Brightspace to start your solution. You need to extract its contents to some work folder on your hard drive.

The starter code/skeleton project is an Eclipse Maven project and you need to import the project as an existing Maven project:

### Import

1. In Eclipse, go to File/Import…/Existing Maven Projects
2. Navigate to your project folder
3. When importing, make sure the **pom.xml** file is checked/selected

Please note that you might need to do the following important steps only if needed:

### Important Steps

1. After opening the project in Eclipse
2. Right-click your **project** and go to **Properties**
3. Go to **Deployment Assembly**
4. Click **Add** and choose **Java Build Path Entries**

5. Click **Next** and choose **Maven Dependencies**
6. Click **Finish**

Remember, if you right-click Maven->Update Project, you have to do the above steps again.

## Starting Assignment 2

You **must** use the skeleton project provided for this assignment to start your solution.

### Your Information

In the folder **src/main/resource**, there is a file called **Bundle.properties** which contains constants used in the UI. Please change the default values for:

```
footer.labsection=Lab Section 301
footer.studentnumber=040123456
footer.studentname=Jane Doe
```

Note: Your name should be as it appears in ACSIS.

### Update the DB

Attached to this assignment on Brightspace, there is a DB script file (**databank2.sql**) which you must run on your DB first.

### Skeleton Project

Project starts with many errors. That is fine. You need to find and fix the issues. There are hints such as TODO markers in the code which can help you.

## Updating the Application – Additional EE Components: Entity Manager and JPA

EntityManager

To connect the DAO to the DB, we need to **inject** a reference (into the DAOImpl) using the `@PersistenceContext` annotation:

- Q1: What is the folder/name of the standard JPA descriptor document (config)?
- Q2: What is the name of the Persistence Unit? (Hint: Look in the 'non-Java' Maven 'resources' folder)

Java Persistence API (JPA)

The StudentPojo model needs additional JPA annotations in order to work. It also needs new member fields to handle the `CREATED`/`UPDATED`/`VERSION` DB columns. The type for the `CREATED`/`UPDATED` member fields should be `java.time.LocalDateTime`.

```
...
@ViewScoped
@Entity(name = "some-name")
@Table(name = "some-name", catalog = "databank", schema = "")
@Access(AcessType.SOME_TYPE)
@EntityListeners({StudentPojoListener.class})
@NamedQuery(name = StudentPojo.STUDENT_FIND_ALL, query = "SELECT something")
...
)
public class StudentPojo implements Serializable {
    private static final long serialVersionUID = 1L;

    public static final String STUDENT_FIND_ALL ="Student.findAll";
...
```

- Q3: What should the `@Entity` name be? What would it be if the name is not set?
- Q4: What should the `@Table` name be? What would it be if the name is not set?
- Q5: What is the JPQL query string for the `@NamedQuery` '`Student.findAll`' that retrieves all students from the database?
- Q6: What are the column names for `firstName`, `lastName`? What would JPA 'think' the DB column names are if the annotations were not set?
- Q7: If annotations are placed on the fields, what should `@Access(AcessType.SOME_TYPE)` be?
- Q8: What type should the member field `version` be?

## Audit Columns

The purpose of the **`CREATED`**/**`UPDATED`** columns is to answer the following questions:

- When was the record *created*?
- When was the record last *updated*?

These columns are referred to as 'audit' columns, they allow you to know when a given row changed. This has obvious benefits in production systems where fraud or security breaches are a concern. Often, DBAs like to solve this problem with *database triggers* or logic that causes a table row to be updated whenever a particular action or event takes place:

```
CREATE OR replace TRIGGER set_create_date_for_student
  BEFORE INSERT ON student
  FOR EACH ROWBEGIN
    -- Update create_date column to current system date
    :new.create_date := SYSDATE;
END;
```

The above **PLSQL** code (**P**rocedural **L**anguage for **SQL**) works only on Oracle databases. Accomplishing the same task across multiple databases can greatly increase the complexity of an application when support of multiple database vendors is required.

We can implement a solution using JPA that works across **all** databases. In the previous section, we discussed adding two new member fields to handle the `CREATED`/`UPDATED` columns. To populate the audit member fields automatically, we use a JPA ‘listener’ that is invoked whenever a particular event occurs. There are seven events (Note: This is a good quizzes/final exam material.):

1. `@PrePersist` – executed before the `EntityManager` *persist* operation is actually executed
2. `@PreRemove` – executed before the `EntityManager` *remove* is actually executed
3. `@PostPersist` – executed after the `EntityManager` *persist* operation (INSERT SQL already committed)
4. `@PostRemove` – executed after the `EntityManager` *remove* operation (DELET SQL already committed)
5. `@PreUpdate` – executed before the UPDATE SQL is executed
6. `@PostUpdate` – executed after the UPDATE SQL is committed
7. `@PostLoad` – executed after an entity has been loaded into the current `@PersistenceContext` or when entity has been refreshed (`EntityManager` *refresh* operation)

```
import javax.persistence.PrePersist;
import javax.persistence.PreUpdate;
...
    @PrePersist
    public void setCreatedOnDate(arg) {

        // TODO
    }
    @PreUpdate
    public void setUpdatedOnDate(arg) {

        // TODO
    }
...
```

The `StudentPojo` class is missing the `@EntityListeners` annotation, it needs to be added. Please refer to the **JSF Regional Inventory 2** sample project discussed during the lecture.

- Q9: What is the argument to `setCreatedOnDate/setUpdatedDate`?

## Updating the Application – JSF, EJB, and JPA

The JSF controller is un-aware that some new EE components and APIs (EntityManager and JPA) are now associated with the StudentDao managed bean, and its API has changed only slightly. The StudentDaoImpl is also simplified with no direct manipulation for the DB using SQL, since all DB operations are delegated to the `EntityManager`.

Our new application uses JPA which needs to be configured and initialized. Luckily, when running inside an EE application server (such as Payara), this is simple:

- In **payara-resources.xml**, specify the Payara JDBC connection pool and its mapping to a JNDI name (already done in Assignment 1, no need to change for Assignment 2).
- In the JPA standard persistence deployment descriptor file (that is, **persistence.xml**) (what was the answer to above **Q1** and **Q2**?), ensure that the entry `<jta-data-source>jndi-name</jta-data-source>` is properly setup.

Note: In the upcoming Assignment 3, we will see how configuring and initializing JPA outside of Java EE is much more complicated!

## Updating the Application – JSF Navigation and SPA

In Assignment 1, the Student Directory application provided CRUD capabilities across three (3) JSF views:

1. `list-students.xhtml`
2. `add-student.xhtml`
3. `edit-student.xhtml`

Even though the application is a simple CRUD list-view application, navigating between views can become quite difficult. Amongst the various controller methods are hard-coded strings such as `"add-student?faces-redirect=true"` and `"list-students?faces-redirect=true"`. What if, for example, a 'Sign In' functionality needs to be added to the application?

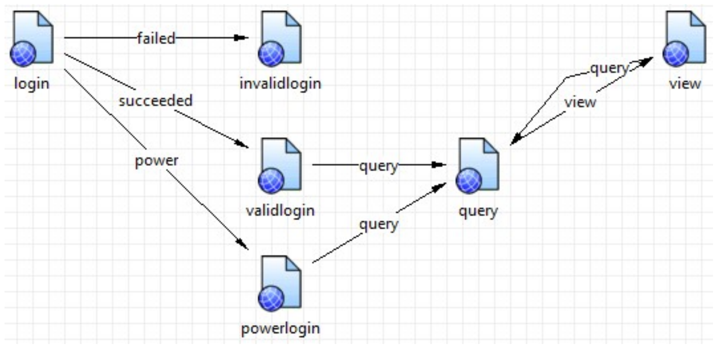


*Figure 1. Credit to Russell Bateman 'A short treatise on JSF' (https://bit.ly/2RToMnQ)*

JSF supports separating controller code from navigation decisions by allowing the developer to specifynavigation rules in the **`faces-config.xml`** file:

```
<navigation-rule>
    <display-name>query</display-name>
    <from-view-id>/query.xhtml</from-view-id>
    <navigation-case>
        <from-outcome>submit</from-outcome>
        <to-view-id>/view.xhtml</to-view-id>
    </navigation-case>
</navigation-rule>
<navigation-rule>
    <display-name>view</display-name>
    <from-view-id>/view.xhtml</from-view-id>
    <navigation-case>
        <from-outcome>query</from-outcome>
        <to-view-id>/query.xhtml</to-view-id>
    </navigation-case>
</navigation-rule>
```

Even with this separation, it is easy to see that the navigation rules for a fully-featured application would become very hard to maintain. This is one of the motivating factors in the move to single-page applications (SPA), where the web app stays on a single page instead of loading new pages from the server, rewriting its content either via direct DOM manipulation or asynchronously using AJAX calls.[1]

In JSF, the primary way to manage this is to ensure that 'outcomes' of actions invoked on various components (i.e. `<h:commandButton action="#{studentController.someOutcome()}"/>`) is to return a `null String`, or the method signature is changed to return `void`. Over on the 'View' side of things, the primary mechanism to help the user interact with the application is to toggle (i.e. hide/show) various on-screen artifacts via their `rendered` attribute (which is present on almost all JSF components), delegating control of the toggle-state to the controller, or in the case of editing an individual row of `<h:dataTable>`, to the model object itself.

- Q10: What must be added to `StudentController` in order to toggle (i.e. hide/show) the 'Add New Student' view?
- Q11: What must be added to `StudentPojo` in order to toggle (i.e. hide/show) editing? If a new member field is added to `StudentPojo`, what about the JPA mapping? Should the state of this member field be stored in the DB? If not, how does one prevent that from happening?

## Updating the Application – JSF `@ViewScoped` Helper Class

In Assignment 1, to hold the "to-be-created"/"to-be-updated" student data, we either use 'spare' fields in `StudentController` or used the (`@Inject`'d) `sessionMap` to hold them. In Assignment 2, we introduce a new class `NewStudentView` to specifically handle this responsibility.

- Q12: What annotation(s) should we add to `NewStudentView`?
- Q13: What field(s) should we add to `NewStudentView`?
- Q14: How is `NewStudentView` used in **index.xhtml**?

[1] Wikipedia (2020). *Single-page application* (retrieved 2020/02/02 https://en.wikipedia.org/wiki/Single-page_application)

## Submitting Assignment 2

To submit your Assignment 2, first you must export your completed project as a zip file. Eclipse has an export function to create an external archive, i.e. a zip file, of your project. Make sure to not include the "target" subfolder when generating the zip file (see screenshots below). You should name your zip file as **<lastname>-<firstname>-Assignment2.zip**, for example, **Smith-John-Assignment2.zip**. You must then upload the correct generated zip file to Brightspace under Assignment 2 (Brightspace/Activities/Assignments/Assignment 2).

Please make sure that your completed project compiles and builds successfully, has good indentation and coding style, not overly complicated, and is efficient. The completed code must demonstrate your understanding of how a JSF and JPA work.

**Note:** Do not forget to upload the correct generated zip file to Brightspace under Assignment 2.

## Demoing Assignment 2

For Assignment 2, please note that demo is <u>not mandatory</u> but <u>optional</u>. However, if you think you have completed your project early (before the due date), you can demo your Assignment 2 to your lab professor during your respective lab session before the due date. The advantage of doing an optional demo for your Assignment 2 is that if the demo is done before the due date, then you will have a chance to fix issues/bugs found during the demo before you submit your completed project on Brightspace. Please note that you are allowed to keep submitting updated versions of your project on Brightspace (before the due date) though only the latest version will be considered your official submission and it will be the one to be graded by your lab professor.

– end –